A Note on Parallel Algorithmic Speedup Bounds

Neil J. Gunther

February 8, 2011

Abstract

A parallel program can be represented as a directed acyclic graph. An important performance bound is the time T_∞ to execute the critical path through the graph. We show how this performance metric is related to Amdahl speedup and the degree of average parallelism. These bounds formally exclude superlinear performance.

1 Computational DAG

A parallel program can be represented as a directed acyclic graph (DAG), where nodes correspond to tasks (or subtasks) and arrows represent control or communication between tasks. Leiserson [1] characterizes the performance of parallel programs by the elapsed time T_1 to execute all the nodes in a DAG (e.g., Fig. 1), and the time T_∞ to execute the *critical path*.

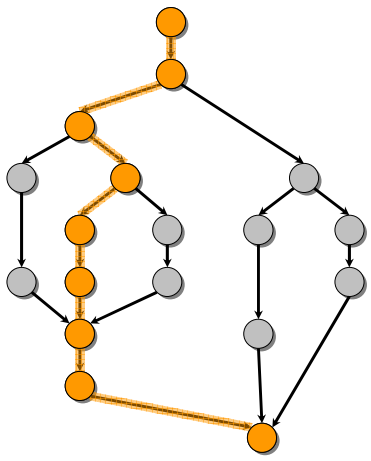

Figure 1: Critical path (*orange*) in an parallel task graph [1]

In project management, a critical path is the sequence of project network activities (e.g., a PERT chart) which add up to the *longest* overall duration. It determines the shortest possible time to complete the project. Any delay of an activity on the critical path directly impacts the planned project completion date (i.e. there is no float on the critical path). A project can have more than one critical path.

The time to execute a program on p processors is T_p and the speedup metric is:

$$S_p = \frac{T_1}{T_p} \tag{1}$$

with computational efficiency:

$$E_p = \frac{S_p}{p} \tag{2}$$

i.e., the average amount of speedup per processor.

2 Performance Bounds

Leiserson [1] claims there are two *lower* bounds on parallel performance for Fig. 1:

$$T_p \geq T_1/p \tag{3}$$

$$T_p \geq T_\infty \tag{4}$$

T_1/p is the reduced execution time attained by partitioning the work (equally) across p processors. Clearly, T_p cannot be less than the time it takes to execute a p-th of the work—the meaning of (3). Similarly, T_p cannot be less than the time it takes to execute the critical path, even if there are an infinite number of physical processors—the meaning of (4).

Substituting (3) into (1):

$$S_p = \frac{T_1}{T_1/p} = p \tag{5}$$

which corresponds to ideal *linear* speedup. In reality, we expect the speedup to be generally sublinear:

$$S_p \leq p \tag{6}$$

Under certain special circumstances speedup may exhibit *superlinear* performance:

$$S_p > p \tag{7}$$

Leiserson excludes (7) on the basis of (3). He also states that because of (4), the *maximum* possible speedup is given by:

$$S_\infty = \frac{T_1}{T_\infty} \tag{8}$$

He calls (8) the "parallelism" and it corresponds to the average amount of work-per-node along the critical path. But what do these bounds really mean?

2.1 Example

Consider an example based on Fig. 1.

Example 1 *Following Leiserson, let's assume for simplicity that each node in the DAG takes just 1 unit of time to execute. Then, the total time to execute the entire DAG on a single processor is* $T_1 = 18$ *time steps.*

Similarly, the critical path contains nine nodes, so $T_\infty = 9$ *and from (8):*

$$S_\infty = \frac{18}{9} = 2 \tag{9}$$

Hence, the maximum possible speedup is 2.

Note, however, that this maximum speedup (8) is not the same as the more familiar Amdahl bound [3, 4, 5]:

$$S_\infty^{\text{Amdahl}} = \frac{1}{\sigma} \tag{10}$$

Equation (10) is the asymptotic form of the Amdahl speedup function [3]:

$$S_p^{\text{Amdahl}} = \frac{p}{1 + \sigma(p-1)} \tag{11}$$

in the limit of an infinite number of processors $p \to \infty$. In Fig. 1, the *serial fraction* (σ) corresponds to 4 single nodes out of 18 total nodes and therefore:

$$S_\infty^{\text{Amdahl}} = \frac{18}{4} = 4.5 \tag{12}$$

which is numerically greater than the "maximum" in (9).

3 Reconciliation

How can we reconcile these various algorithmic speedup metrics?

3.1 Average Parallelism

Theorem 1 *Leiserson's* S_∞ *is identical to the average parallelism [2, 3] defined as:*

$$A = \frac{W}{T} \tag{13}$$

where W *is the total amount of work (expressed in cpu-seconds, for example) and* T *is the total parallel execution time.*

Proof 1 *Calculate W using the following procedure:*

1. *Start at the top of the DAG*
2. *At each level where there are nodes, draw a horizontal line through them*
3. *On each horizontal row calculate the time-node product for each node*
4. *Sum all the time-node products on each row to get W*

which can be written symbolically as:

$$W = \sum_{i=1}^{\text{depth}} t_i \times n_i \tag{14}$$

For Fig. 1 we obtain:

$$W = (1 \times 1) + (1 \times 1) + (1 \times 1) + (1 \times 3) + (1 \times 4) + (1 \times 4) + (1 \times 2) + (1 \times 1) + (1 \times 1) \tag{15}$$

or $W = 18$. The value of T can be obtained be simply adding together all the time factors in the products of (15), i.e., $T = 9$, since $t_i = 1$ and there are nine terms. Applying (13): $A = 18/9 = 2$, which is identical to (9). Thus, $S_\infty \equiv A$. ■

Remark 1 *This is consistent with Leiserson's definition of S_∞ as the average amount of work-per-node along the critical path. See Section 2. Other examples of calculating average parallelism are presented in Ref. [6].*

3.2 Superlinear Performance

Finally, we can see how Leiserson excludes superlinear performance on the basis of bound (3).

Theorem 2 *The bound (3) is equivalent to any computational DAG compressed to depth one.*

Proof 2 *In Fig. 1, such node compression is equivalent to having all 18 nodes positioned on the same horizontal row. Since it is not possible to squash the DAG any flatter, the best possible speedup corresponds to distributing those 18 nodes simultaneously onto $p = 18$ processors. This bound is identical to ideal linear speedup (5), i.e., $S_p = 18$.* ■

Corollary 1 *From (2), linear speedup corresponds to an efficiency $E_p = 18/18 = 1$.*

Superlinear speedup (7) corresponds to an efficiency $E_p \geq 1$. One way this might be observed is to run the work in Fig. 1 successively on $p = 1, 2, 3, \ldots$ processors. The speedup for small-p would be inferior to that for large-p, so the scaling would appear to become better than linear. However, this apparent improvement is just an artifact of choosing the wrong baseline to establish linearity in the first place.

References

[1] C. E. Leiserson, "Multithreaded Programming in Cilk," MIT, 2008 lecture-1.pdf

[2] D. L. Eager, J. Zahorjan, E. D. Lozowska, "Speedup Versus Efficiency in Parallel Systems," IEEE Transactions on Computers, Volume 38 Issue 3, March 1989

[3] N. J. Gunther, *The Practical Performance Analyst*, iUniverse, 2000

[4] N. J. Gunther, "A New Interpretation of Amdahl's Law and Geometric Scalability," arXiv:cs/0210017 2002

[5] N. J. Gunther, "A General Theory of Computational Scalability Based on Rational Functions," arXiv:0808.1431 2008

[6] N. J. Gunther, USL Fine Point: Sub-Amdahl Scalability February 4, 2011